计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.190500132

一种自适应优化松弛量的装箱算法

杨 婷 罗 飞 丁炜超 卢海峰
华东理工大学信息科学与工程学院 上海 200237
(Real_yting@hotmail.com)

摘 要 装箱问题是物流系统和生产系统中的一个经典而重要的数学优化问题。装箱指把一系列物品按照一定顺序放进具有固定容量的箱子中,并最小化所使用的箱子数量,以最大限度地获取装箱问题的近似最优解。然而,现有的装箱算法存在明显的缺陷。遗传算法计算量过大,甚至无法求出所需解,启发式算法无法处理极端值问题,而现有的改进算法即使在引入松弛量的情况下,也极易陷入局部最小值。文中提出的 Adaptive-MBS 算法采用自适应权重来改进原有方法,即允许方法有一定的松弛量,并具有捕捉物体样本空间随时间变化的直觉,以使用更好的松弛量策略来装箱。Adaptive-MBS 算法首先以当前箱子为中心,使用 Adaptive_Search 搜索算法迭代找到适合箱子容量的集合中所有物体的子集,Adaptive_Search 搜索算法不要求完全装满箱子,而是允许箱子具有一定的松弛量,在训练过程中根据当前状态的变化,实现自动地调整松弛量,在找到完全填满箱子的子集后迭代至下轮搜索直至遍历完成。该方法不易陷入局部最优,具有较强的发现全局最优解的能力。文中使用装箱问题中经典的 BINDATA 和 SCH_WAE 数据集进行实验,结果表明,数据集中多达 991 例问题可以通过 Adaptive-MBS 算法得到最优解。在没有求解出最优解的实例上,所提算法也在所有对比算法上具有最低的相对偏移量百分比。数值实验结果表明,相较于其他经典的装箱算法,Adaptive-MBS 算法有更好的效果,其收敛速度也显著优于其他算法。

关键词:装箱问题;自适应权重;启发式算法;松弛量;全局最优解
中图法分类号 TP301.6

Bin Packing Algorithm Based on Adaptive Optimization of Slack

YANG Ting,LUO Fei,DING Wei-chao and LU Hai-feng
Department of Computer Science and Engineering,East China University of Science and Technology,Shanghai 200237,China

Abstract The bin packing problem is a classical and important mathematical optimization problem in logistics system and production system. A series of items are put into bins with fixed capacity in a certain order,and the number of bins used is minimized to obtain the approximate optimal solution of the bin packing problem to the greatest extent. However,the existing bin packing algorithms have obvious defects. Genetic algorithm has too much computation,and even can't find the required solution. Heuristic algorithm can't deal with the extreme value problem. And the existing improved algorithm will easily fall into the local minimum even if the slack is introduced. The proposed Adaptive-MBS algorithm uses adaptive weights to improve the original method. Specifically,the method is allowed to have a certain amount of slack,and has the intuition of capturing the change of object sample space with time,so as to use a better slack strategy to pack. The Adaptive-MBS algorithm first uses the current bin as the center and uses the Adaptive_Search algorithm to iteratively find a subset of all objects in the set suitable for the bin capacity. In the Adaptive_Search algorithm,the bin is not required to be completely filled,but is allowed to have a certain amount of slack. In the training process,the slack is automatically adjusted according to the change of the current state,and after finding the subset that is completely filled,the subset is iterated to the next round of search until the traversal is completed. This method is not easy to fall into local optimum and has strong ability to find global optimum. In this paper,the BINDATA and SCH_WAE data sets in the packing problem are used for experiments. The results show that 991 cases in the data set can be optimized by Adaptive-MBS algorithm. In the case where the optimal solution is not found,the proposed algorithm has the lowest relative offset percentage in all comparison algorithms. Numerical experiments show that compared with other classic bin packing algorithms,Adaptive-MBS algorithm has better effect and its convergence speed is significantly better than other algorithms.

Keywords Bin packing problem,Adaptive weight,Heuristic algorithm,Slack,Global optimal solution

到稿日期:2019-05-22 返修日期:2019-08-13 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金面上项目(61472139);华东理工大学 2017 年教育教学规律与方法研究项目(ZH1726107)
This work was supported by the National Natural Science Foundation of China(61472139),and Research Project of Education and Teaching Law and Method of East China University of Science and Technology in 2017 (ZH1726107).
通信作者:罗飞(luof@ecust.edu.cn)

1 引言

装箱问题(Bin Packing Problem,BPP)是经典的离散组合最优化问题,在离散、有限的数学结构上,寻找满足给定条件并使其目标函数达到最大或最小的解。在装箱问题中,不同体积的物体必须以最小化所用箱数的方式装入容量为 c 的有限数量的容器中。容器的大小是固定和已知的,目标是尽量减少容器的数量,使总成本最小化。BPP 在实践中有许多应用,例如,多处理器任务调度、并行处理、多媒体存储和切割库存问题等。因此,如何快速高效地寻找装箱策略具有充分的现实意义。有效的装箱算法旨在缩短计算时间、降低总装箱成本、提高资源利用率。

目前常用于装箱问题的算法包括启发式算法[1]、元启发式算法[2]、分支定界算法[3]等。随着深度学习的兴起,深度强化学习[4-5]也在解决装箱问题上崭露头角。由于 BPP 在计算复杂性理论中是一个组合的 NP-hard 问题[6],随着物体数量的增加,可行解呈指数增加。它的决策问题(决定物体是否放进指定数量的容器)是 NP-complete[7]的。因此,装箱算法的优化是一个非常热门的研究方向。

MBS(Minimum Bin Slack)[8]算法是广泛使用的装箱算法之一。该算法每次采用字典搜索程序(Lexicographic Search Optimization Procedure,简称 L 算法)寻找最优子集。MBS 算法遍历待装箱的物体,每个步骤都旨在找出最适合当前箱子容量的物体的集合。当箱子被完全装满或再也没有其他物体可以装入给定箱子时,迭代过程终止。文献[8]同时证明了该算法在装箱物体的重量和小于或等于两倍的箱子容量时,一定可以获得最优解。

然而,现有的大多数基于 MBS 的方法都有缺陷。这些算法在训练过程中由于忽略了对全局物体样本空间的探索,极易陷入局部最优解。在迭代训练后期,由于局部最优解的偏差不断累积,全局最优解空间不断偏移,造成算法结果与最优解具有显著差异,这导致了装箱结果不佳。

现有的基于 MBS 的方法的训练策略可能是短视的,这些训练策略通常趋于选择当前的最优选择,最终无法学习完整的物体样本空间。但是彻底地探索样本空间以实现全局上更好的性能又难以实现。理想的装箱算法应该平衡以上问题,因此本文提出了自适应松弛量策略。

具体来说,本文为了解决 MBS 经典装箱算法易陷入局部最优解的问题,提出了一种自适应松弛量装箱算法。与现有研究相比,本文算法能获得更优解与更好的时间复杂度。本文的主要贡献如下:

(1)在 MBS 算法中引入自适应最小松弛量,构建自适应搜索策略,以更好地指导 MBS 算法的实现;

(2)所提出的 Adaptive-MBS 算法可以学习到良好的自适应松弛量值,使得该算法不易陷入局部最优。将该算法应用于数据集的大规模样例时,其由于具有更强的发现全局最优解的能力,因此可以探索到更好的容器选择策略。

2 相关工作

20 世纪 70 年代初期 NP-hard[9]问题完备性建立后,一维装箱问题就得到了广泛关注。目前,国内外已有较多的相关研究。由于装箱问题是一个 NP-hard 问题,因此学者大多采用启发式算法来找到最接近最优解的近似解方案。

Coffman 等[10]首次全面回顾了各种启发式算法[11]。FF(First Fit),NF(Next Fit),BF(Best Fit)和 WF(Worst Fit)等都是已知的在线启发式方法。在线启发式方法是按照物体到达的顺序对其装箱。离线启发式算法则要求事先对物体的大小以非递增顺序进行排序,然后应用 FF 或 BF,常用的算法有 BFD[12](Best Fit Decreasing),FFD[13](First Fit Decreasing)和 WFD(Worst Fit Decreasing)。适应算法是优先在已装入物体的箱子中装入物体,找不到合适的箱子后才启用新的箱子。这种优先对箱子进行操作的方法,保证了每个到达的物体在当前情况下总能找到一个箱子去容纳该物体,但这也带来了一个问题:无法保证当前情况下容纳物体的箱子是适合于该物体的,即当前解不是较优解。

Gupta 等[8]提出了一种新型的启发式算法 MBS,MBS 总是以箱子为中心,尽最大可能找到填满箱子的物体集合。这种方式保证了当前情况下容纳该物体的箱子一定适合于该物体,但是该方法的顺序选择策略经常短视地关注输入空间的局部区域,导致无法实现对物体数据空间的准确估计,因此极易陷入局部最优解。

为解决上述问题,一些方法被提出。Fleszar 等[14]提出了一种混合算法,它基于 MBS 的元启发式算法引入可变邻域搜索的概念。BPP 的变邻域搜索算法是基于移动的。移动被定义为将一个物体从当前的箱子转移到另一个箱子,或者在它们各自的当前的存储箱中交换一对物体。但是,该方法由于计算量巨大未被广泛接受。

Martello 等[15]在书中引入了简化程序和精确算法来描述简单的启发式方法和装箱下界。精确算法如分支定界算法、贪婪技术、启发式算法和元启发式算法都是解决离线装箱问题的常用方法。Zhang[16]针对一维组合装车问题,建立了线性混合整数规划和分支定界算法模型,并提出了基于贪婪技术的启发式算法。遗传算法[17-18]后期因为缺失粒子多样性,容易出现早熟收敛。当问题复杂时,其不能及时地利用训练过程中的反馈信息,而且由于涉及大量个体的计算,导致计算时间过长,结果的稳定性差。精确算法的一个问题是在问题规模很大时,不能在合理的时间内找到最优解。

近年来,Dokeroglu 等[19]提出用岛并行分组遗传算法优化一维装箱问题,并行的遗传算法有助于不同种群同时向不同的方向进化,通过产生更多的个体和世代来产生更高质量的解决方案。Loh 等[20]提出了一种权重退火(Weight Annealing,WA)程序,依靠权重退火的概念,通过在搜索空间的各个部分产生扭曲来扩展和加速搜索。Negre 等[21]借助整数线性规划的方法寻找问题的最优解。Hu 等[4]提出用深度强化学习方法来求解装箱问题。

与上述经典的一维装箱算法相比,本文方法可以获得更好的表现。

3 问题描述

经典一维装箱问题要求把一系列物品按照一定顺序包装

成多个相同大小和形状的容量为 c 的箱子，其目的是尽量减少装 n 个物品所需的箱子数量[15]。本文研究的经典一维装箱问题的形式化描述如下。

给定一组待装载物体的集合 $L=\{a_1,a_2,\cdots,a_s\}$，物体的大小为 $s(a_i)$，将其装入容量为 C 的箱子中。假设箱子的数量是无限的，且每个箱子的容量 $C>0$。寻求一种分拆方法 A，将 L 拆分成互不相交的子集 B_j，使得 B_j 所有物品大小之和不超过箱子容量，求满足这一要求的最小整数 m。则有：

$$\begin{cases} L=B_1\cup B_2\cup\cdots\cup B_m & (1)\\ \forall i,a_i\in B_j,0<\sum\limits_{i\in bin(k)}s(a_i)\leqslant C & (2)\\ \forall j,1\leqslant j\leqslant m,s(B_j)\leqslant C & (3)\\ \forall a_i\in L,\exists a_i\in B_j & (4)\\ \min m \end{cases}$$

显然，对于任何物体 $a_i\in L$，B_j 中总存在一个装填位置。即所有 L 中的物体全部完成装箱。

4　解决装箱问题的启发式算法

4.1　经典的启发式算法

经典的离线启发式装箱算法首先建立物体的降序集合，之后按照物体的到达顺序对其进行装箱。NFD(Next Fit Decreasing)算法是常见的启发式装箱算法，该算法在当前箱子的剩余容量小于装箱物体质量时，启用下一个箱子。FFD(First Fit Decreasing)算法按容量将物体以非增序排列，并将其连续装箱。每次将物体放入索引号最小的箱子中，同时不可超过箱子所能容纳物体的容量限制，并不断迭代至遍历所有实例。WFD(Worst Fit Decreasing)算法在对物体降序排列之后，在不超出箱子容量限制的条件下，每次选择最小负载的箱子来获得大规模物体实例的最优解。BFD(Best Fit Decreasing)算法在 WFD 算法的基础上进行调整，为了更加稳健地选择策略和更加精确地求解，每次选择最大负载的箱子。

Gputa 等[8]提出了基于最小化松弛量的(MBS)启发式算法。MBS 用一个搜索程序迭代找到适合箱子容量的物体子集，子集保持最小松弛量。在每次迭代过程中，如果算法找到一个完全填满箱子容量的子集，则此回合的搜索终止，直到遍历完所有集合中的物体，迭代终止。可以看出，MBS 总是以箱子为中心，尽最大可能找到填满箱子的物体集合。

Gputa 等同时证明了该算法在装箱物体的重量和小于或等于两倍的箱子容量时，装箱竞争比会达到 1。MBS 每次采用字典搜索程序寻找最优子集，具体如算法 1 所示。

算法 1　MBS 算法

Input：t_i for i=1,…,n;C;k=1;s=n

Step1　使用搜索方法 L，寻找应分配给箱子 k 的物体集合 S_k。令 $\sigma=(\sigma(1),\sigma(1),\cdots\sigma(s))$为未分配箱子的物体编号。

Step2　If $\sigma=\varnothing$，Go to Step3；Else，k=k+1，Go to Step1。

Step3　将 $S=(S_1,S_2,\cdots,S_k)$逐个放入箱子(1,2,…,k)，最小松弛量为 $kC-\sum\limits_{i=1}^{n}t_i$。

MBS'[14]算法是 MBS 算法的改进。在 MBS 算法的基础上，引入“种子物体”的概念，即固定第一个物体，每次从第二个物体开始装箱。由于其是离线的装箱算法，物体大小以非升序方式排列，显然该算法每次都选择集合中最大的物体作为“种子物体”，在此基础上对剩下的物体使用 MBS 装箱。

Perturbation MBS'[14]算法在现有的装箱策略上添加扰动。该算法选择剩余空间最大的箱子作为种子，其中物品被放入集合中等待装箱，其他箱子按照剩余空间大小递减的顺序排列，依次从集合中取物体放入新箱子，迭代一定次数后，若箱子数目未有减少或箱子数目达到下限，则算法停止。

Relaxed MBS'[14]算法提出 L 搜索程序的终止条件由 $s=0$ 变为 $s\leqslant allowable\ slack$，松弛量每次以 $v=0.5\%*c$ 大小增加，增加的限制次数为 $\min\{40,c/v\}$。若达到装箱下界，则程序终止。

4.2　Adaptive_MBS 算法

寻找应该分配给容器的物体集合的搜索方法 L 是不同 MBS 算法变体的核心。原始的搜索方法 L 配置简单的选择策略，但是因为其执行时间太长，需要用高效的方法来删减搜索空间，同时针对原始方法易陷入局部最优的问题，本文提出了 Adaptive_MBS 算法。Adaptive_MBS 算法的主要思路是通过自动适应物体样本空间变化的松弛量来改进选择策略。

简单地在装箱算法中使用松弛量能够使算法在一定程度上依靠随机性跳出局部最优陷阱，但它没有学习样本空间的精确分布，因此对算法策略的影响具有较大的随机性。因此，本文使用松弛量的一个见解是在装箱问题的训练迭代期间通过样本空间的变化对松弛量添加多样性的变化。然而，太多的不确定性会使算法不稳定，而太多的多样性会导致策略偏向更多的随机选择。为了在迭代过程中自动捕捉样本空间的变化并选择高质量的装箱策略，松弛量需要理解容器与剩余物体在数据空间上的分布。在本文算法中，决定程序终止条件的松弛量被设置为自动随样本空间的变化而改变的变量，它能够在当前样本空间情况下学习好的策略来选择高质量的装箱顺序。本文将松弛量的变化过程表述为选择决策问题跳出局部最优解的关键。本文方法引入自适应权重 ω，它通过装箱迭代的轮次 t 来驱动，具有捕捉物体样本空间随时间变化的直觉。该算法中，松弛量基于自适应权重 ω 在一个动态变化的范围内产生，自适应于每一个物体的装箱过程。具体做法是，在装箱的每一步骤中，容器每增加一个物体，算法就更新自适应权重，使得在每一轮迭代中，*slack* 在一个新适应的范围内随机产生，从而跳出在局部最优停滞的窘境。Adaptive_MBS 算法的描述如算法 2 所示。

算法 2　Adaptive_MBS 算法

Step1　t->1，初始化自适应权重 ω。

Step2　产生随机松弛量 slack∈(0，min_value * ω)。

Step3　While：还有物体未装箱，使用搜索方法 Adaptive_Search，寻找应分配给箱子 k 的物体集合 S_k，否则，end While。

Step4　将 $S=(S_1,S_2,\cdots,S_k)$逐个放入箱子(1,2,…,k)。

自适应权重是 Adaptive-MBS 算法的核心参数，对算法性能有重要影响。Adaptive-MBS 算法通过在每一轮迭代中实时监测装箱状态，并根据装箱状态动态调整各物体的惯性权重来实现自适应权重，从而消除迭代过程中 *slack* 为定值的不良影响，平衡全局搜索与局部搜索。自适应权重的更新公式为：

$$w=w_{max}-\frac{t(w_{max}-w_{min})}{s} \tag{5}$$

其中，w_{max} 和 w_{min} 是惯性权重系数的最大值和最小值，其典型取值为 $w_{min}=0.4$，$w_{max}=0.9$；t 表示每轮迭代中装箱物体在原物体队列中所在的位置，即当前迭代装到了第几个物体；s 表示装箱物体总数。本文所提的搜索方法 Adaptive_Search 的描述如算法 3 所示。

算法 3 搜索方法 Adaptive_Search

Input: t_i for i=1,…,s; C; k=1; $\sigma=(\sigma(1),\sigma(1),\cdots\sigma(s))$，其中 $t_{\sigma(1)}\geqslant t_{\sigma(2)}\geqslant\cdots\geqslant t_{\sigma(s)}$; $\alpha=0$; j=1; $\pi=(\pi(1),\pi(2),\cdots,\pi(j))=(\sigma(1),\sigma(2),\cdots,\sigma(j))$; $\omega=1$

Output: $S=(S_1,S_2,\cdots,S_k)$

Step1 产生随机松弛量 slack∈(0, min_value * ω)，进入 Step2。

Step2 如果($0\leqslant C-P_\pi\leqslant$ slack)，令 $S_k=\pi$，进入 Step7；否则进入 Step3。

Step3 找到 q 使 $\sigma(q)=\pi(j)$。如果 $P_\pi<C$，令 j=j+1，按照 $w=w_{max}-\frac{t(w_{max}-w_{min})}{s}$ 更新权重 ω，进入 Step4；否则进入 Step6。

Step4 如果 $P_\pi>\alpha$，令 $\alpha=P_\pi$，$S_k=\pi$，进入 Step5。

Step5 如果 q<s，令 $\pi_j=\sigma(q+1)$，进入 Step1；否则进入 Step6。

Step6 如果 j=1，进入 Step7；否则，令 j=j−1，按照 $w=w_{max}-\frac{t(w_{max}-w_{min})}{s}$ 更新权重 ω，找到 q 使 $\sigma(q)=\pi(j)$，进入 Step5。

Step7 把 S_k 中的物体放置到第 k 个箱子中。

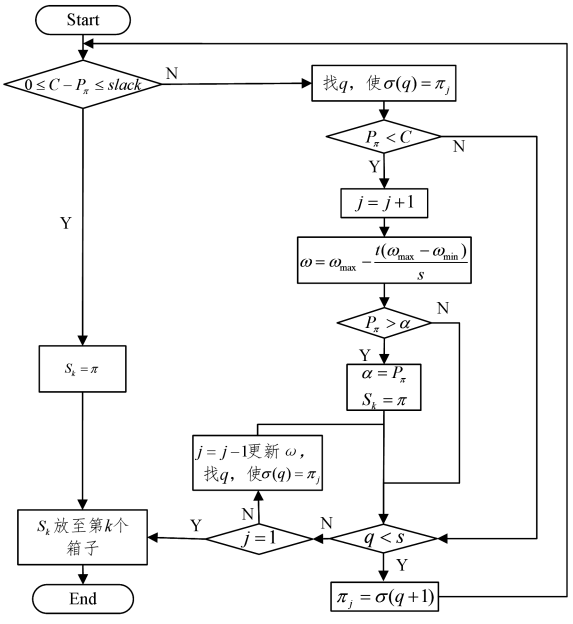

图 1 Adaptive_MBS 算法的流程图

Fig. 1 Flow chart of Adaptive_MBS algorithm

5 实验设计与结果分析

5.1 实验环境

实验在 Ubuntu16.04 环境下进行，配置为 CPU24 Intel(R) Xeon(R) CPU E5-2620 @ 2.6GHz，将 Pycharm 作为开发环境。

5.2 评价指标

5.2.1 竞争比

一维装箱问题要求把物体放进固定容量的箱子中，并最小化所使用的箱子数目。近似算法的有效性通过竞争比[22]来评估。介绍竞争比之前，首先引入最优解的概念。OPT 定义为每个装箱实例的最优解所包含的箱子数，在实际解决装箱问题的过程中，最优耗费一般是未知的，因此，一般用理论最优解代替 $OPT(\sigma)$，它存在一个下限值[23-26]。由于装箱条件的限制，每次装箱所使用的箱子数量不可能小于物体总重量与单个箱子容量的比值，即：

$$OPT(\sigma)\geqslant\left\lceil\frac{\sum_{i=1}^{k}t_i}{C}\right\rceil \tag{6}$$

其中，t_i 代表第 i 个物体的重量，C 为箱子容量。竞争比的定义如下：

$$Competitive\_Ratio=\frac{SOL}{OPT} \tag{7}$$

其中，SOL 代表近似算法所使用的箱子数，OPT 定义为每个装箱实例的最优解所包含的箱子数。竞争比为 1，意味着算法找到了最优解。显然，目标是使竞争比尽可能接近于 1，即算法所找到的装箱策略越来越接近于最优解。

5.2.2 实现最优解个数及其偏移量

通过对比各算法来实现已知最优解的数量以评估各算法的优劣，是一个很直观的评价方式。但是，在遇到极端数据集时，存在的一种可能性是各算法均不能实现实例的已知最优解。因此，引入偏移量的概念，所谓偏移量是指各算法实现的解与最优解的偏差百分比。通过比较各算法离最优解的偏差(用 Gap 表示)来评估算法性能，其定义为：

$$Gap=\frac{SOL\text{-}OPT}{OPT} \tag{8}$$

5.3 实验数据集

本文使用 BINDATA[27] 和 SCH_WAE[28] 两个数据集。Scholl 等[27]提出的 BINDATA 数据集包含 3 组数据，第一组数据(Bin1data)由 720 个实例组成，其中 704 个实例目前得到了最优解。第二组数据(Bin2data)由 480 个实例组成，有最优解的实例有 477 个。在 Bin1data 和 Bin2data 集合中，物体的数量从 50 到 500 不等。第三组数据(Bin3data)有 10 个实例，每个实例都有 200 个重量均匀分布在(20000,35000)区间的物体，这些物体被装入大小为 100000 的容器中。这组实例非常难以求解，其中只有 3 个已知的最优解。SCH_WAE 数据集来自 Schwerin[28] 和 Waescher。这个数据集包含 200 个实例，并给出了目前已知的最优解。

将本文结果与参考解进行比较，参考解是最优解，或者是不知道最优值的已知下界(1410 例中有 26 例)，详细信息如表 1 所列。

表 1 问题实例的相关信息

Table 1 Information about problem instances

数据集	实例数	物体重量	箱子容量(c)	物体个数(n)
Bin1data	720	[1,100]	{100,120,150}	{50,100,200,500}
Bin2data	480	[1,700]	1000	{50,100,200,500}
Bin3data	10	[20000,35000]	100000	200
SCH_WAE	200	[150,200]	1000	{100,120}

5.4 实验结果

为了验证 Adaptive-MBS 算法的有效性，本文分别在 BINDATA 和 SCH_WAE 数据集上，将其与经典适应算法、

MBS算法与MBS'算法进行了实验对比。每个数据集的各个实例中,每个类别的物体数相同,物体的多样性由弱到强,因而能够很好地测试在不同多样性下算法解决问题的性能。实验结果如表2－表8所列。其中,表2－表5依次列出了8种算法在两个数据集上的装箱结果。

表2　Bin1data数据集上的实验结果

Table 2　Experimental results on Bin1data dataset

算法	实现已知最优解	未实现已知最优解	平均竞争比
NFD	0	720	1.3502
FFD	546	174	1.0497
WFD	442	278	1.0537
AWFD	163	557	1.0663
BFD	547	173	1.0497
MBS	252	468	1.0645
MBS'	633	87	1.0471
Adaptive-MBS	**661**	**59**	**1.0459**

表3　Bin2data数据集实验结果

Table 3　Experimental results on Bin2data dataset

算法	实现已知最优解	未实现已知最优解	平均竞争比
NFD	59	421	1.1271
FFD	236	244	1.0315
WFD	213	267	1.0336
AWFD	1	479	1.0806
BFD	236	244	1.0315
MBS	125	355	1.0829
MBS'	247	233	1.0264
Adaptive-MBS	**291**	**189**	**1.0175**

表4　Bin3data数据集的实验结果

Table 4　Experimental results on Bin3data dataset

算法	实现已知最优解	未实现已知最优解	平均竞争比
NFD	0	10	1.1711
FFD	0	10	1.0739
WFD	0	10	1.0739
AWFD	0	10	1.0865
BFD	0	10	1.0739
MBS	0	10	1.0594
MBS'	0	10	1.0721
Adaptive-MBS	0	10	**1.0469**

表5　SCH_WAE数据集的实验结果

Table 5　Experimental results on SCH_WAE dataset

算法	实现已知最优解	未实现已知最优解	平均竞争比
NFD	1	199	1.0622
FFD	1	199	1.0614
WFD	1	199	1.0614
AWFD	0	200	1.1069
BFD	1	199	1.0614
MBS	25	175	1.0574
MBS'	32	168	1.0513
Adaptive-MBS	**39**	**161**	**1.0425**

从实验的装箱结果可以看出,本文提出的Adaptive_MBS算法在两个数据集上均实现了最多的已知最优解实例,同时平均竞争比也是最低的。本文算法的竞争比接近1,说明其找到的装箱策略越来越接近于最优策略。

从表6、表7可以看出,对总共1410个基准问题进行实验,其中70.3%(991例)的问题可以通过Adaptive-MBS得到最优解。且对于未实现最优解的实例,本文方法较最优解的相对偏移量百分比最低。对比其他经典算法,本文提出的Adaptive-MBS算法具有较强的寻找全局最优解的能力。

表6　算法在各数据集上较最优解的相对偏移量百分比

Table 6　Relative offset percentage of optimal solution of algorithms on each data set

(单位:%)

算法	Bin1data	Bin2data	Bin3data	SCH_WAE
NFD	35.03	12.71	17.12	6.22
FFD	4.98	3.15	7.39	6.15
WFD	5.38	3.37	7.39	6.15
AWFD	6.64	8.06	8.66	10.69
BFD	4.98	3.15	7.39	6.15
MBS	6.46	3.08	5.95	5.17
MBS'	4.71	2.98	7.21	5.10
Adaptive-MBS	**4.59**	**1.75**	**5.05**	**4.25**

表7　Adaptive-MBS实现已知最优解的总实例数

Table 7　Total number of instances of optimal solution by Adaptive-MBS

数据集	实例数	实现已知最优解	需额外箱子实例数
Bin1data	720	661(91.8%)	59
Bin2data	480	291(60.6%)	189
Bin3data	10	0	10
SCH_WAE	200	39(19.5%)	161
总计	1410	991(70.3%)	419

由表8可知,Adaptive_MBS算法的时间复杂度是指数型的,其中执行k次的Adaptive_Search搜索方法的时间复杂度为$O(2^n)$,本文通过自适应物体样本空间变化的松弛量来改进选择策略,并没有引入额外的时间开销。因此,Adaptive_MBS算法的时间复杂度为$O(2^n)$。基于测试用例,本文算法在数据集的小规模样例中可以充分发挥竞争比优势;在数据集的大规模样例中,在计算能力充足时,尽管时间成本会相应提高,但是由于算法具有更强的发现全局最优解的能力,同样可以获得更好的装箱竞争比。另外,从表8可知,Adaptive-MBS累计实现最优解991例,显著优于其他算法。由此,本文所提算法是有效的。

表8　各算法累计实现的最优解及平均竞争比

Table 8　Cumulative optimal solution and average competition ratio of each algorithm

算法	累计实现最优解	累计平均竞争比	时间复杂度
NFD	60	1.1776	$O(n\log n)$
FFD	783	1.0541	$O(n\log n)$
WFD	656	1.0556	$O(n\log n)$
AWFD	164	1.0850	$O(n\log n)$
BFD	784	1.0541	$O(n\log n)$
MBS	402	1.0660	$O(2^n)$
MBS'	912	1.0492	$O(2^n)$
Adaptive-MBS	**991**	**1.0382**	$O(2^n)$

结束语　本文提出了用于一维装箱问题[29]的Adaptive-MBS算法。该算法根据容器与物体样本空间分布自适应地动态改变算法行为参数,将自适应权重值引入松弛量系数中,由装箱迭代的轮次t自动驱动,不依赖多余的计算量。特别地,该方法具有捕捉物体样本空间随时间变化的直觉,可以在大规模数据集上探索到更好的选择策略。在几个基准数据集上的实验结果表明,所提方法显著优于其他经典的装箱方法。同时,如何对松弛量进行其他改进,以更好地寻找全局最优解将是未来装箱问题的重要研究方向。

参 考 文 献

[1] DE ALMEIDA R, STEINER M T A. Resolution of 1-D Bin Packing Problem using Augmented Neural Networks and Minimum Bin Slack[C]// Latin America Congress on Computational Intelligence (LA-CCI). IEEE, 2015: 1-6.

[2] KAAOUACHE M A, BOUAMAMA S. Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud[J]. Procedia Computer Science, 2015, 60(1): 1061-1069.

[3] RODRIGO N P, DAUNDASEKERA W B, PERERA A I. One-dimensional bin-packing problems with branch and bound algorithm[J]. International Journal of Discrete Mathematics, 2018, 3(2): 36.

[4] HU H, ZHANG X, YAN X, et al. Solving a new 3d bin packing problem with deep reinforcement learning method[J]. arXiv: 1708.05930, 2017.

[5] DUAN L, HU H, QIAN Y, et al. A Multi-task Selected Learning Approach for Solving 3D Flexible Bin Packing Problem[J]. arXiv: 1804.06896, 2018.

[6] KORF R E. A new algorithm for optimal bin packing[C]// Proceedings of the EighteenthNational Conference on Articial Intelligence. AAAI Press, 2002: 731-736.

[7] VAZIRANI V V. Approximation algorithms [M]. Springer Science & Business Media, 2013: 10-19.

[8] GUPTA J N D, HO J C. A new heuristic algorithm for the one-dimensional bin-packing problem[J]. Production Planning & Control, 1999, 10(6): 598-603.

[9] HARTMANIS J. Computers and intractability: a guide to the theory of NP-completeness (michael r. garey and david s. johnson)[J]. Siam Review, 1982, 24(1): 90.

[10] COFFMAN J, GAREY M, JOHNSON D S. Approximation algorithms for NP-hard problems. [M]. Boston: PWS Publishing, 1997: 46-93.

[11] DOSA G. Encyclopedia of Algorithms [M]. New York: Springer, 2014: 10-19.

[12] NGUYEN-DUC T, QUANG-HUNG N, THOAI N. BFD-NN: best fit decreasing-neural network for online energy-aware virtual machine allocation problems[C]// Proceedings of the Seventh Symposium on Information and Communication Technology. ACM, 2016: 243-250.

[13] JOHNSON D S. Near-optimal bin packing algorithms[D]. Cambridge: Massachusetts Institute of Technology, 1973.

[14] FLESZAR K, HINDI K S. New heuristics for one-dimensional bin-packing[J]. Computers & operations research, 2002, 29(7): 821-839.

[15] MARTELLO S, TOTH P. Lower bounds and reduction procedures for the bin packing problem[J]. Discrete applied mathematics, 1990, 28(1): 59-70.

[16] ZHANG J J. Conbinatorial packing probleming: Models and Algorithms[D]. Shanghai: Shanghai Jiao Tong University, 2012.

[17] SRIDHAR R, CHANDRASEKARAN M, SRIRAMYA C, et al. Optimization of heterogeneous Bin packing using adaptive genetic algorithm[C]// IOP Conference Series: Materials Science and Engineering. IOP Publishing, 2017.

[18] QUIROZ-CASTELLANOS M, CRUZ-REYES L, TORRES-JIMENEZ J, et al. A grouping genetic algorithm with controlled gene transmission for the bin packing problem[J]. Computers & Operations Research, 2015, 55: 52-64.

[19] DOKEROGLU T, COSAR A. Optimization of one-dimensional bin packing problem with island parallel grouping genetic algorithms[J]. Computers & Industrial Engineering, 2014, 75: 176-186.

[20] LOH K H, GOLDEN B, WASIL E. Solving the one-dimensional bin packing problem with a weight annealing heuristic[J]. Computers & Operations Research, 2008, 35(7): 2283-2291.

[21] HIFI M, NEGRE S, WU L. Hybrid greedy heuristics based on linear programming for the three-dimensional single bin-size bin packing problem[J]. International Transactions in Operational Research, 2014, 21(1): 59-79.

[22] BALOGH J, BÉKÉSI J, DÓSA G, et al. The optimal absolute ratio for online bin packing[C]// Proceedings of the twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms. Philadelphia: Society for Industrial and Applied Mathematics, 2014: 1425-1438.

[23] KHANAFER A, CLAUTIAUX F, TALBI E G. New lower bounds for bin packing problems with conflicts[J]. European Journal of Operational Research, 2010, 206(2): 281-288.

[24] CLAUTIAUX F, DELL'AMICO M, IORI M, et al. Lower and upper bounds for the Bin Packing Problem with Fragile Objects [J]. Discrete Applied Mathematics, 2014, 163: 73-86.

[25] BALOGH J, BÉKÉSI J, GALAMBOS G, et al. Lower bound for 3-batched bin packing[J]. Discrete Optimization, 2016, 21: 14-24.

[26] BALOGH J, BÉKÉSI J. Semi-on-line bin packing: a short overview and a new lower bound[J]. Central European Journal of Operations Research, 2013, 21(4): 685-698.

[27] SCHOLL A, KLEIN R, JÜRGENS C. Bison: A fast hybrid procedure for exactly solving the one-dimensional bin packing problem[J]. Computers & Operations Research, 1997, 24(7): 627-645.

[28] SCHWERIN P, WÄSCHER G. The bin-packing problem: A problem generator and some numerical experiments with FFD packing and MTP[J]. International Transactions in Operational Research, 1997, 4(5/6): 377-389.

[29] LÓPEZ-CAMACHO E, TERASHIMA-MARÍN H, ROSS P. A hyper-heuristic for solving one and two-dimensional bin packing problems[C]// Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation. Dublin: ACM, 2011: 257-258.

YANG Ting, born in 1996, postgraduate. Her main research interests include cloud computing and bin packing problems.

LUO Fei, born in 1978, Ph.D, associate professor, postgraduate supervisor, is a member of China Computer Federation. His main research interests include cloud computing, and distributed computing.